Transitioning from Human to Agent-Based Role-Players for Simulation-Based Training

Robert G. Abbott[✉], Christina Warrender, and Kiran Lakkaraju

Sandia National Laboratories, Albuquerque, NM 87111, USA
rgabbot@sandia.gov

Abstract. In the context of military training simulation, "semi-automated forces" are software agents that serve as role players. The term implies a degree of shared control – increased automation allows one operator to control a larger number of agents, but too much automation removes control from the instructor. The desired amount of control depends on the situation, so there is no single "best" level of automation. This paper describes the rationale and design for Trainable Automated Forces (TAF), which is based on training by example in order to reduce the development time for automated agents. A central issue is how TAF interprets demonstrated behaviors either as an example to follow specifically, or as contingencies to be executed as the situation permits. We describe the behavior recognizers that allow TAF to produce a high-level model of behaviors. We assess the accuracy of a recognizer for a simple airplane maneuver, showing that it can accurately recognize the maneuver from just a few examples.

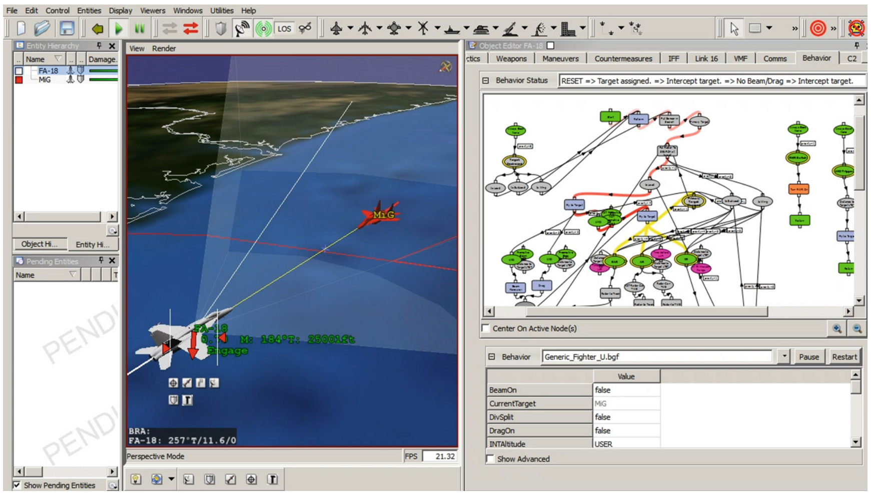
Keywords: Learning by example · Training by demonstration · Software agents

1 Introduction

When automating a task that has previously been performed by a person, recordings of human performance data can provide examples of desired behaviors. Our current domain is fighter combat, where student pilots fly against "enemy" role-players who emulate anticipated threat tactics. The tactics can be recorded in an airplane instrumented with position tracking equipment, or from a flight simulator. This paper presents our recent work on Trainable Automated Forces (TAF), which is a system to simulate tactical behavior based on recorded examples.

In the context of Live/Virtual/Constructive training [1], TAF is designed to create a Constructive entity from a Live or Virtual (i.e. human in the loop) example. The output of TAF is a set of inputs for the Next Generation Threat System (NGTS), which is a constructive entity simulator that focuses on fighter tactics [2] (Fig. 1). NGTS simulates both the physics and behaviors of entities, but TAF focuses exclusively on the behaviors. The design of TAF assumes that NGTS is capable of producing the desired

D.D. Schmorrow and C.M. Fidopiastis (Eds.): AC 2015, LNAI 9183, pp. 551–561, 2015.
DOI: 10.1007/978-3-319-20816-9_53



Behavior Status: RESET => Target assigned. => Intercept target. => No Beam/Drag => Intercept target.

Behavior: Generic_Fighter_U.bgf

	Value
BeamOn	false
CurrentTarget	MiG
DivSplit	false
DragOn	false
INTAltitude	USER

Fig. 1. The next-generation threat system (NGTS) simulates fighter tactics. On the right is a portion of the decision logic for the white aircraft on the left. TAF addresses the difficulty of developing this logic.

tactics if it is programmed correctly, but programming the tactics can be difficult and time-consuming. The objectives of TAF are:

1. Minimize the time and cost of creating new behavior models.
2. Create a fully realized, executable NGTS entity from recordings that do not explicitly capture the goals of the pilot or details non-visible behavior such as turning on sensors or jammers and making radio calls.
3. Interpret the example behavior literally or generally as desired by the user, inferring which actions are to be replicated, and which arose by chance and should only be executed if necessary.
4. Strike a balance between open-ended behaviors that respond realistically to a wide range of situations and aren't overly predictable, yet don't distract from the training objectives by being erratic.
5. Fall back to a reasonable general-purpose behavior model when there are no relevant examples in the set of scenario recordings.
6. Learn from a small number of training examples; flight recordings are expensive to obtain.
7. Produce behavior models that are human-readable, allowing the model to be manually corrected or tailored if necessary.

The key to balancing these goals is leveraging the subject matter expertise that is built into NGTS, re-combining and parameterizing the provided behaviors. The alternative would be to build models based completely on flight recordings, ignoring the existing physics and behavior models in NGTS. However, unless vast amounts of example behavior data are available, a software agent based solely on observed behaviors will be limited. The range of agent behaviors will be limited to behaviors that occurred within the training data, and the agent will take inappropriate actions, because a large number of examples are required to distinguish casual effects from spurious correlations. Moreover, some aspects of the behavior cannot be learned by observation because they are not observable to the system, e.g. the employment of sensors and emitters by a fighter pilot.

2 Related Work

Programming by demonstration (PBD) is the approach of programming an agent by replicating the actions of a person, whose actions are recorded in a simulator or instrumented environment (e.g. an airplane that records its position over time). PBD is a natural approach to progressively automating a task that was formerly performed by a person, such as role-playing for training scenarios. PBD was originally used for tasks such as assembly-line robotics in which the robot would carry out the same sequence of steps each time. Most of this research has been done in the field of robotics and targets low-level sensor/motor actions [3]. As more complex tasks were undertaken, generalizing the behavior to accommodate different situations became increasingly important. Rather than replicating actions specifically, the agent must infer the goals implied by the example, and select and refine actions as necessary to accomplish those goals.

Goals are necessarily domain-specific, so PBM must be adapted to each new domain with a set of possible goals and means to attain them. Specifying a set of goals compromises the original intent of PBD to reduce manual programming, but ensures that the set of goals are consistent with how human experts understand the domain. This is crucial if humans are to monitor to agent's actions and sometimes influence them for reasons external to the system, such as instructors using TAF to train students.

3 TAF Design and Rationale

This section describes the design of TAF, and how the design supports the objectives listed in the introduction (Fig. 2).

3.1 TAF Knowledge Base

The TAF Knowledge Base (KB) is the key to mapping desired agent behaviors to the NGTS inputs that elicit that behavior. It contains numerous examples of

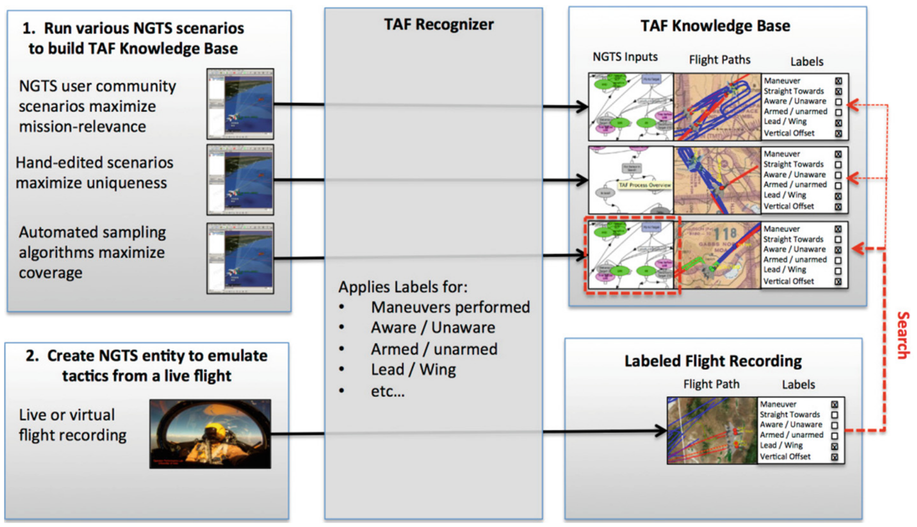

1. Run various NGTS scenarios to build TAF Knowledge Base
NGTS user community scenarios maximize mission-relevance
Hand-edited scenarios maximize uniqueness
Automated sampling algorithms maximize coverage
2. Create NGTS entity to emulate tactics from a live flight
Live or virtual flight recording
TAF Recognizer
Applies Labels for:
Maneuvers performed
Aware / Unaware
Armed / unarmed
Lead / Wing
etc...
TAF Knowledge Base
NGTS Inputs
Flight Paths
Labels
Maneuver
Straight Towards
Aware / Unaware
Armed / unarmed
Lead / Wing
Vertical Offset
Labeled Flight Recording
Flight Path
Labels
Search

Fig. 2. TAF. During development, the knowledge base is populated with NGTS scenario executions that establish the entity behaviors resulting from various NGTS inputs. Then TAF can generate NGTS entities whose behaviors match those found in a flight recording. The TAF recognizer is the basis for this matching.

scenarios executed in NGTS that include agents using various settings. These include the behavior rules assigned to the agent, parameters for the behavior rules, and the scenario in which the agent was embedded. The KB is analogous to the experience accumulated by a human user of NGTS who experiments with various settings and observes the results to develop an understanding of what to expect from the software.

The mapping from NGTS inputs to agent behaviors may change as the NGTS software is revised, so it is important that TAF be able to re-learn this association in an efficient manner. We have developed TAF training software that automatically executes an NGTS run for each of a set of inputs which may be very large, so TAF can re-learn each new version of NGTS.

The behavioral repertoire of TAF is constrained by the KB examples – it can only take actions contained in its knowledge base. A benefit of the KB-limited approach (vs. generating inputs before or during scenario execution) is that the KB can be audited to remove unwanted examples. However, the parameter space of NGTS is practically infinite - far too large to explore exhaustively. Therefore the selection of scenarios for the KB is very important. To date we have used three sources. **NGTS user community scenarios** are developed manually by NGTS users to meet specific end-user needs, so they are full of realistic situations and behaviors of great use to TAF. On the other hand, re-combining elements from them will not allow TAF to author scenarios much different than users already have, so additional sources are needed. We have developed a number of **hand-edited** scenarios that each differ significantly from each other, and from previously existing scenarios. Scenarios from **automated sampling algorithms** allow TAF to explore beyond human-specified entities. These are easy to generate in large numbers and can potentially allow TAF to output any type of behavior of which NGTS is capable, including behaviors not envisioned by NGTS developers. The risk is of generating large numbers of behaviors that are unrealistic, irrelevant, or redundant. Our approach is to use scenarios from the user community and our own hand-edited scenarios as the basis for subsequent modification by the sampling algorithm.

Once each scenario is run, it is inserted into the KB. The KB example includes the paths of all entities and the NGTS inputs used to produce them, such as the version of the NGTS software used and the behavior rules and settings assigned to the entity. Then the TAF Recognizer is applied to recognize actions in the example scenario execution.

3.2 TAF Recognizer

The TAF Recognizer applies semantic labels to traces in the KB, and in flight recordings. The granularity of the labels determines how much TAF will abstract the example scenarios when matching them to flight recordings. For example, a trajectory could be recognized either as a series of specific maneuvers (e.g. "turn left 30°, proceed ahead for 25 NM, then turn left 15°…") or at a more abstract, goal-oriented level ("e.g. Lead Pursuit on target aircraft"). Labels are tested from general to specific, and processing of mutually exclusive labels stops with the first match. However, not all labels are mutually exclusive, and some recognizers use the output of those applied beforehand. For example, several recognizers look at the position of the aircraft relative to each group of enemies, where groups are identified by one of the early recognizers.

TAF recognizers are implemented as feature extractors in the RBBML framework [1]. In the RBBML framework, a *model* is a Java class that defines methods to derive a set of features to derive and store in the database. For example the Intercept Geometry model samples the positions of two entities at regular distance intervals and then computes their turn rates, and the range and aspect angle from each to the other. This model also has receptors for range categories Near/Medium/Far, which are not used in matching, but are consulted after a match is found to determine the range at which the entity will maneuver. A *model instance* is a model populated with examples that exemplify a particular behavior, and specifies which model features are relevant to identifying the behavior. For example the Beam Maneuver model instance contains examples of that maneuver flown by a human pilot and by an NGTS entity and selects the features *target aspect* and *threat turn rate*. The Intercept Geometry model can be trained with different examples to recognize different maneuvers.

Recognizing Groups of Entities. Fighters normally work together, often flying in formation. The basic unit of organization is a *group* of two or more fighters. A constructive simulation of fighter tactics must capture this organizational structure because the roles of two aircraft (*lead* and *wing*) are somewhat different. The roles must be made explicit for NGTS, since otherwise the lead and wing would not stay in formation. However, flight recordings do not specify the formation hierarchy; it must be recognized from spatial information.

The TAF algorithm for recognizing groups is based on two aircraft being "usually near" each other. This is complicated by the fact that aircraft have separate lifetimes (either may exist while the other does not), and that aircraft in the same group may occasionally separate for a period of time. For each pair of aircraft, TAF samples the positions at a regular interval (1 s), and tallies the samples in which the pair of aircraft are near (the distance is less than a threshold of 5 nautical miles using the Haversine formula) vs. far (the distance exceeds the threshold, or only one of the two aircraft currently exist). The two aircraft are considered "usually near" if the ratio of near to far instances exceeds a threshold (at least 80 % of all samples are near). Next, each aircraft is placed into a separate group. Two groups are considered near if any aircraft in one group is "usually near" any aircraft in the other group. The groups, which initially contain a single aircraft each, are merged with all other near groups, until no group is near any other (Fig. 3).

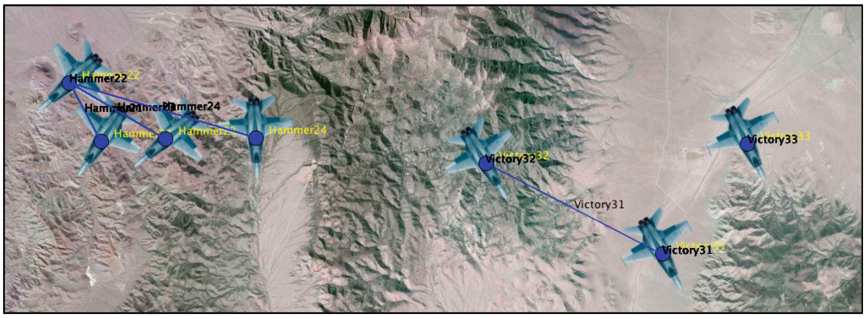


Fig. 3. Six aircraft are assigned to three groups. Note that the grouping does not correspond precisely to the groups implied by the manually-designated callsigns, because victory 33 deviates from victory 31 later in the scenario. Formation-flying is typical in tactical flight, and grouping them reduces the combinatorial complexity of identifying interactions among aircraft.

The leader of each group is recognized by computing the mean velocity vector of members of the group at each timestep. The position of each group member is projected onto the velocity vector, and the member with the highest median projected value is designated the leader.

A limitation of this approach is that group membership and leadership are static – each aircraft belongs to only a single group over time. The advantage of this approach is that it avoids discontinuities in the spatial relations between groups that would occur from re-assigning an aircraft from one group to another. However the results at a point in time may appear incorrect because the group assignment is dominated by the rest of the scenario.

Recognizing Maneuvers. A simple maneuver (e.g. a flying a loop) is a relatively fixed path through space over time relative to a starting position. In combat, maneuvers are

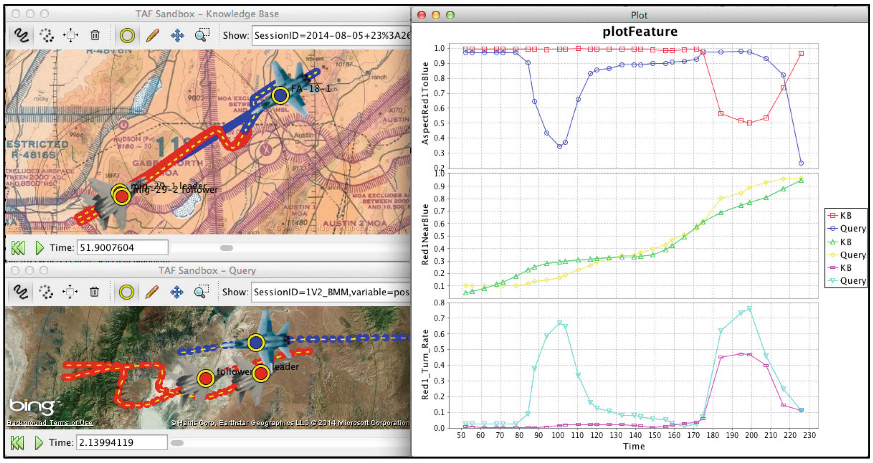


Fig. 4. A recognizer for the beam maneuver. Flight recordings (bottom) are searched for an example of the maneuver (top) on the basis of time-varying spatial features such as range, aspect, and turn rate.

generally executed relative to other aircraft, and the maneuver is characterized by a prescribed path through the state space of relative positions and angles. For example, in a beam maneuver the pilot flies perpendicular to the attacker so as to avoid moving either towards or away from the attacker. If the attacking aircraft or missile has a relatively simple Doppler radar, this maneuver may cause the radar difficulty in distinguishing the aircraft from stationary objects on the ground.

In TAF, maneuvers are sampled from KB examples or flight recordings. For the beam maneuver (Fig. 4), the aspect transitions from 1 (towards) to 0 (perpendicular) then back to 1. The turn rate of the aircraft is also matched to ensure that it, and not its target, is performing the maneuver.

TAF searches for maneuvers by computing the time-varying spatial relations between each group of fighters. The position of a group is defined as the position of the lead aircraft in the group. The rate of execution of a maneuver may vary somewhat, so TAF uses the dynamic time warping algorithm [2] to match spatio-temporal trajectories that vary only in rate of execution.

Spatio-temporal matching always yields approximate matches, so the quality of match is also recorded on a scale from 100 (perfect match) to 0 (not considered a match). The percentage is used to place different recognizers on a comparable scale.

3.3 TAF Search

In TAF the goal of search is to find examples of NGTS inputs that resulted in an entity exhibiting the desired behaviors, and secondly to rank the results order of match quality.

The first step (search) is relatively trivial because the recognizer assigns tags to KB entries, and search simply retrieves all entries with *at least* the required tags. Earlier iterations of TAF [3] compared the path of the flight recording to each KB example individually, but this was unnecessarily specific (since it is the *meaning* or label of the path that is important, not its precise trajectory) and too computationally intensive to support a large knowledge base. Labeling the KB examples and the flight path individually resolves this problem.

The second step is ranking, which is based on the quality of fuzzy labels. For example, maneuvers never *precisely* match the ideal. Recognizers may assign a "percent match" value between 0 and 100. If present, they are used to order the KB matches.

3.4 Flight Recording

To create an NGTS entity, the user loads a flight recording into TAF and selects the example entity. When the flight recording is read in, TAF applies the recognizer to it. A user interface is supplied so the user can correct the recognized behaviors, or change the labels used in the search so the resulting NGTS entity differs from the example in some aspects.

The set of TAF behavior labels must be compatible with richness or sparsity of the source of the flight recording. For example, a manually-controlled entity in NGTS will result in a highly detailed flight recording without any sensor noise. A virtually-piloted entity recorded from a network simulation (commonly the HLA protocol [4]) will also be noise-free, but will generally only have information about the externally-visible attributes of the aircraft and pilot behavior, with no explicit information about goals. A flight recording from a live source will have sensor noise and dropouts, but (depending on the source) may have detailed information about onboard systems. However, the 'black box' from a single aircraft is not sufficiently informative to model tactics, since tactical flight is generally performed relative to other aircraft, ground targets, etc.

4 Evaluation of TAF Behavior Recognition

This section describes an assessment we performed on the TAF Behavior Recognizer for the beam maneuver that was described in the previous section. The behavior recognizer is a key element of TAF, because it determines the similarity between a flight recording to be emulated and the repertoire of behaviors in the TAF Knowledge Base. We collected flight recordings, identified beam maneuvers independently both manually and with the TAF Recognizer, and then compared the results.

We collected a set of examples of the beam maneuver in the context of a larger mission that was executed repeatedly. This data set included executions of the maneuver both by human pilots, and by NGTS entities. In the case of NGTS entities, we performed recognition on the maneuver using only positional information, disregarding whether the maneuver had been triggered by the NGTS Beam Maneuver

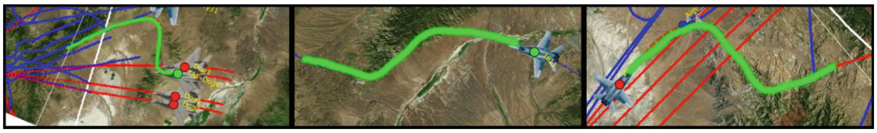


Fig. 5. Sections of flight paths that, to varying degrees, resemble an ideal beam maneuver in which the aircraft turns to place an oncoming threat directly to the left or right. TAF assigned a match quality of 99 % (left), 55 % (middle), and 34 % (right).

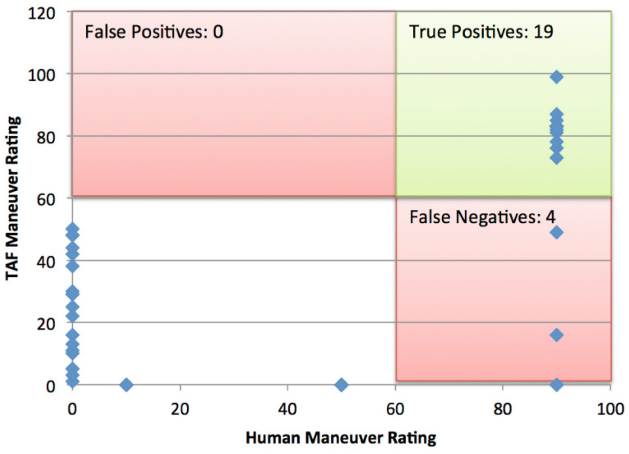


Fig. 6. Ratings of 23 candidate instances of the beam maneuver in a set of flight recordings

command. (Modeling NGTS behaviors is of interest because they may be controlled by a human operator manually issuing commands, and these operators are to be modeled in addition to actual pilots.) We examined 77 scenario executions, each of which included 32 airplanes, for a total of 2,464 trajectories.

By examining these trajectories the human rater identified 36 candidate examples of the beam maneuver. Each candidate maneuver was assigned a quality score between 1 and 100, with 100 being ideal, and 1 deemed only marginally similar to the ideal. Note that there is no “ground truth” (beyond the human rater) of whether a beam maneuver was performed – a pilot might try to perform the maneuver and be interrupted, or simply make a mistake. Textbook definitions of maneuvers [7] *may* include tolerances for relevant angles etc. but human raters, and TAF, recognize degrees of success (Fig. 5).

We then ran the TAF recognizer on the same data set. TAF also assigns a quality score, and identified 41 candidate matches. Matches identified by the human and TAF were considered to refer to the same event if they were performed by the same aircraft during time windows that coincided. The probability of matching by chance was low, since fewer than 1 in 60 trajectories contained a candidate example, and a typical example covered less than 10 % of the duration of a single trajectory (80 s vs. 900 s).

The human and TAF ratings of maneuvers identified are shown in Fig. 6. The correlation between human and TAF ratings is 0.63. The correlation is brought down by several possible matches identified by TAF (in the lower-left of the figure). These matches are not necessarily a problem since they are assigned a low quality by TAF, and could be automatically suppressed. With a decision boundary of 60 % quality, TAF obtains 19 true positives, 0 false positives, and 4 false negatives. Of the false negatives (maneuvers recognized by the human rater but not detected by TAF), some are corner cases such as a maneuver performed against a threat that was destroyed before the maneuver was completed. It should be noted that a group of human raters would not have 100 % agreement, although we do not have data on inter-rater reliability.

5 Conclusion and Next Steps

The current capability of TAF matches flight recordings to pre-existing models of behavior that (with appropriate parameters) will exhibit the desired behaviors in the appropriate contexts. In addition, TAF directly adopts a few specific settings (such as initial aircraft and heading) from the flight recording. Our plan is to expand this capability to combining novel sequences of actions that directly model observed behaviors from a flight recording. In the current version of TAF, novel behaviors must be generated a priori and inserted into the knowledge base. This approach is not scalable to all conceivable, valid combinations of behaviors. Our efforts are now directed towards discriminating between planned actions (goals) and actions that arose due to happenstance (contingencies).

Acknowledgement. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

References

1. Schnell, T., Postnikov, A., Hamel, N.: Neuroergonomic assessment of simulator fidelity in an aviation centric live virtual constructive (LVC) application. In: Schmorrow, D.D., Fidopiastis, C.M. (eds.) FAC 2011. LNCS, vol. 6780, pp. 221–230. Springer, Heidelberg (2011)
2. Hildreth, B., Linse, D.J., Dicola, J.: Pseudo six degree of freedom (DOF) models for higher fidelity constructive simulations. In: AIAA Modeling and Simulation Technologies Conference and Exhibit, Hoholulu, Hawaii (2008)
3. Abbott, R.G.: The relational blackboard. In: Behavior Representation in Modeling and Simulation, Ottawa, Ontario, Canada (2013)
4. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD 2012), New York, USA (2012)
5. Abbott, R.G., Lakkaraju, K., Warrender, C.: Semi-automated construction of adversarial agents for trainable automated forces. In: AAMAS, Paris (2014)
6. Kuhl, F., Weatherly, R., Dahmann, J.: Creating Computer Simulation Systems: An Introduction to The High Level Architecture. Prentice Hall, Upper Saddle River (1999)

7. Naval Air Training Command, Flight Training Instruction: Air to Air Intercept Procedures Workbook (CNATRA P-825), Department of the Navy, NAS Corpus Christi, TX, USA (2010)
8. Sigaud, O., Peters, J.: Abstraction levels for robotic imitation: overview and computational approaches. In: Sigaud, O., Peters, J. (eds.) From Motor Learning to Interaction Learning in Robots. SCI, vol. 264, pp. 1–12. Springer, Heidelberg (2010)